Configurable Function Implementing System and Digital to Analogue Converters

The present invention relates to configurable function implementing systems such as vector dot product multipliers, and to digital to analogue converters. The digital to analogue converters of the present invention are applicable in particular, though not necessarily, for use in vector dot product multipliers.

Vector dot product multiplication is a fundamental operation in discrete-time signal processing, and is stated mathematically as:

$$y = \sum_{k=1}^{K} w_k x_k \qquad (1)$$

where w_k and x_k are the kth element of the weights and input vector respectively, y is the output and K is the size of the input vector.

By allowing for a discrete time dimension, denoted by the subscript n in the output and the weight, equation (1) may be restated for vector matrix multiplication (VMM) operations:

$$y_n = \sum_{k=1}^{K} w_{k,n} x_k \qquad (2)$$

where the output vector y_n is obtained sequentially in time according to the periodic change in weight $w_{k,n}$. Together, vector dot product and VMM are at the core of numerous applications, such as those performing finite impulse response filters, discrete Fourier transforms and the discrete cosine transforms.

When referring to system implementations of vector dot product and VMM operations, the following properties may be relevant:

Programmability - refers to the ability to change at run-time the coefficients of the individual weight elements so as to change the functionality of the device.

Scalability

a) of precision - refers to the ability to change at run-time the SNR characteristic of the system;

b) of complexity - refers to the ability to change at run-time the size of the operation

Reconfigurability - refers to the ability to change at run-time the number of operations that are carried out in parallel in a given network, and hence reconfigurability embodies and extends the definition of scalability of complexity.

In purely digital systems, digital signal processors (DSPs) are typically used to implement vector dot product and VMM operations. However, such implementations are often power hungry and inefficient due to the large number of multiplications involved. With a DSP, a multiply operation is performed bit-wise, and is multi-staged, requiring clock frequencies that are several times higher than the signal frequency.

Vector dot product and VMM operations are not limited to the digital domain and examples of analogue and mixed-signal vector dot product and VMM cells exist. Such operations are described for example in:

[1] R. Genov and G. Cauwenberghs, "Charge-Mode Parallel Architecture for Vector Matrix Multiplication," Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on, vol. 48, pp. 930-936, 2001;

[2] V. A. Pedroni, "Error-compensated analog cells for vector multiplication and vector quantization," Circuits and Systems II: Analog and Digital Signal [3] Processing, IEEE Transactions on, vol. 48, pp. 511-519, 2001; and

[3] T. Y. Lin and A. J. Payne, "Programmable analogue vector-matrix multiplier," Electronic Letters, vol. 38 pp. 1-2, 2002.

In the implementation described in [1], while the charge-mode multiply operation is performed in the analogue domain, the digital representation is embedded within the architecture, where inputs are presented in bit-serial fashion and matrix elements are stored locally in bit-parallel form. In this approach, the multiplication is performed through a series of Boolean AND operations implicit in the charge transfer operation. A more 'analogue' approach is undertaken in [2], where the inputs, weights and outputs are represented by continuous analogue variables, and multiplication is achieved by exploiting the square-law characteristic of MOS transistors in saturation. [3] takes a different approach, where the inputs and weights modulate the phases of the inputs and weights current respectively. A translinear multiplier coupled with low-pass filtering is used to obtain the output.

In these prior art 'analogue' approaches, the topology of the systems are fixed at design-time and the systems lack the reconfigurability referred to above. For such approaches, programmability (i.e. definition of an array of discretely weighted coefficients) may be obtained by the integer scaling of a modulating parameter, such as the area of a capacitor or resistor, and the coefficient is given by the ratio of that parameter value to the unit value.

In general, digital approaches are superior to analogue approaches in terms of noise and precision. The two performance metrics are interrelated as the precision of the system dictates the upper-bounds of the signal-to-noise ratio for a given dynamic range. In digital approaches, the precision of the system is determined by the "width" of the bus. In analogue approaches where the weights are programmed digitally, the precision of the system may be limited by noise and errors, such as those due to matching.

As already noted, a component of a vector matrix multiplier operating in the mixed analogue-digital signal domain is likely to be a digital to analogue converter (DAC). DACs play a fundamental and necessary role in bridging the divide between the quantised data manipulated in digital space and the continuous signals that the real world interacts with. However, for mobile and portable devices necessitating low power consumption and small chip area, DACs consume a substantial part of the power budget. As the precision required by an application increases, physical problems such as matching, power supply sensitivity and other issues become more critical, placing a lower limit on the minimum quantisation step size. Constraints on chip area and power consumption then place an upper limit on the resolution (i.e. number of input bits) of the device. Hence, for a given minimum quantisation step, a lower resolution DAC consumes less power since it has a correspondingly lower full scale swing.

A precise, low power, programmable and reconfigurable method of performing vector operations such as vector dot products and VMM, as well as scalar operations such as polynomial functions for function approximation, for signal processing and computation is needed to meet the growing demand for autonomous, body-worn, lifestyle and sensor interface products. In any such method, optimisation of DAC performance is likely to be key.

According to a first aspect of the present invention there is provided apparatus for converting an M-bit digital signal into an analogue signal, the apparatus comprising:

means for mapping the M-bit digital signal to first and second digital values, so that the ratio of the first to the second digital value is equal to or approximates the value of the M-bit digital signal;

first and second digital to analogue converters, the first digital to analogue converter having an input for receiving said first digital value and the second digital to analogue converter having an input for receiving said second digital value; and

circuit means coupled to the analogue outputs of the digital to analogue converters for dividing one of the analogue outputs by the other, and for providing the result to an output.

The bit length of the first and second digital values is less than that of the M-bit digital signal. Preferably, the bit length N of the first digital value is the same as that of the second digital value, although this need not be the case.

The bit precision which can be achieved by combining two linear digital to analogue converters according to the present invention is greater than that which can be achieved by using one of these converters alone. Furthermore, the power consumed by the two linear (N-bit) converters is generally less than that which would be consumed by a single digital to analogue converter having (2N-1) bits. Another advantage of embodiments of the present invention is that the quantisation steps are smaller for small signal levels than they are for high signal levels. Signal to (quantisation) noise ratios are therefore improved for small signal levels.

Preferably, said means for mapping comprises a memory storing a look-up table, the look-up table containing fractional values and respective first and second value pairs, such that the ratio of a first and second value is equal to the corresponding fractional value. The means for mapping further comprises means for looking-up the table to find the closest fractional approximation to the M-bit digital signal, and for identifying the corresponding first and second values.

In certain embodiments of the invention, the apparatus comprises means for compressing said M-bit digital signal by some factor A. The compressed M-bit digital signal is passed to the means for mapping. Said circuit means comprises means for scaling the result of said division by the factor A. As the signal to noise ratio is higher for small signal levels, the signal to noise ratio across the whole dynamic range is improved.

Preferably, said circuit means is a translinear multiplier.

According to a second aspect of the present invention there is provided a method of converting an M-bit digital signal into an analogue signal, the method comprising:

mapping the M-bit digital signal to first and second digital values, so that the ratio of the first to the second digital value is equal to or approximates the value of the M-bit digital signal;

applying said first and second digital values to inputs of the first and second digital to analogue converters respectively; and dividing the analogue output of one of the digital to analogue converters by the other, and providing the result to an output.

According to a third aspect of the present invention there is provided apparatus which is configurable to evaluate a function, the apparatus comprising:

a plurality of scaling elements, each scaling element having a first input for receiving an analogue input signal, a second input, and an output;

control means for generating a digital weight for one or more of said scaling elements, and having output means for applying generated weights to the second inputs of respective scaling elements;

output means having a plurality of inputs coupled to outputs of respective scaling elements to receive scaling products therefrom, a plurality of outputs selectively coupled to respective inputs, and means for selectively coupling inputs or outputs together, the control means being coupled to the output means for effecting the selective coupling.

Said scaling elements may be multiplication elements, division elements, or elements configurable to perform either multiplication or division.

The scaling elements may be purely analogue devices in which case digital to analogue conversion means is provided for converting the digital weights into analogue weights. Alternatively, the scaling elements may be mixed digital and analogue devices in which case the digital weights may be applied directly to the scaling elements.

In one embodiment of the invention, the apparatus is configurable to operate as a vector dot product multiplier.

Embodiments of the present invention provide a vector dot product multiplier which can be reconfigured both in terms of the weights applied to the scaling elements and in terms of the combination of elements used to perform a given scaling operation. For example, by appropriately selecting the connections formed in the output means, the apparatus may be configured to perform two or more multiplication operations in parallel, each operation using a subset of the plurality of scaling elements.

Preferably, said control means is a microprocessor or microcontroller capable of periodically reprogramming one or more said digital weights.

Preferably, each scaling element comprises a digital to analogue converter (DAC) having its digital input coupled to the second input of the element so as to receive a digital weight from the control means. More preferably, the DAC receives at a control input thereof said analogue input signal, the output of the DAC being coupled to the output of the scaling element to provide at the output, the multiplication product.

Preferably, the output means comprises a first plurality of switches for selectively coupling neighbouring inputs of the output means together, and a second plurality of switches coupling inputs of the output means to respective outputs.

Outputs of the output means may be selectively coupled to provide feedback to the scaling elements. For example, for each scaling element, summing means may be provided for summing an initial weight with a value present at one of these said outputs, with the result being applied as the digital weight to the second input of the scaling element. The selection of outputs for feedback is preferably controlled by said control means.

According to a fourth aspect of the present invention there is provided a method of evaluating a polynomial function using apparatus according to the above third aspect of the present invention, the method comprising:

factorising the function so as to put it into a form containing nested multiply and accumulate terms;

applying the function variable to first inputs of at least certain of the scaling units and applying function constants to second inputs of at least certain scaling units; and

configuring the apparatus so that components of each multiply and accumulate term are evaluated by respective scaling elements and summed by the output means, with each intermediate sum being fed back to a scaling element evaluating a component of the next order multiply and accumulate term.

For a better understanding of the present invention and in order to show how the same may be carried into effect reference will now be made, by way of example, to the accompanying drawings, in which:

Figure 1 illustrates schematically a MSRMAC system;

Figure 2 illustrates a multiplier cell of the system of Figure 1;

Figure 3 illustrates a reconfiguration bridge of the system of Figure 1;

Figure 4 illustrates schematically, a simplified digital to analogue conversion system;

Figure 5 illustrates in more detail the system of Figure 4;

Figure 6 illustrates an alternative digital to analogue conversion system;

Figure 7 illustrates another alternative digital to analogue conversion system;

Figure 8 shows the possible quanta for digital to analogue conversion system having a base precision of 4-bits;

Figure 9 is a set of plots comparing the performance of the system of Figure 5 with that of a conventional linear digital to analogue converter; and

Figure 10 illustrates a multiplying digital to analogue conversion system obviating the need for a separate multiplier;

Figure 11 is a set of plots comparing the performance of the system of Figure 9 with that of a conventional linear digital to analogue converter;

Figure 12 illustrates schematically apparatus for evaluating a polynomial function;

Figure 13 illustrates schematically apparatus for evaluating a third order polynomial function; and

Figure 14 illustrates switches of a reconfiguration bridge of the apparatus of Figure 13.

The mixed-signal reconfigurable multiply and accumulate (MSRMAC) system which will now be described comprises an array of analogue input ports that feed an array of multiplier cells that are configurable by a microprocessor. The outputs of the multiplier cells are applied to inputs of a reconfiguration bridge (R-BRIDGE) which is also configurable by the microprocessor. It will be appreciated that whilst the system may be implemented using discrete components or may be integrated into a chip operating as a standalone component, the preferred implementation is within an integrated system designed to perform some predefined or programmable function(s) making use of vector operations such as vector dot product or vector matrix multiplication or scalar operations such as in the generation of polynomial functions for the purpose of function approximation or otherwise.

Referring to Figure 1, a MSRMAC system is illustrated which comprises an array of analogue input ports 1 which feed a one-dimensional array of "analogue" multiplier cells 2 with respective analogue signals x_k. Digital weights w_k are applied to weighting inputs 3 of respective multiplier cells 2 by a microprocessor 4. Each cell 2 comprises a digital to analogue converter (DAC) coupled to the cell input 3. The DAC may be a conventional, linear current-output DAC such as a current-steering DAC, or a "rational" DAC such as is described in detail below.

The multiplication functionality of the cells 2 may be implemented by any circuit means which produces an output which is proportional to the multiplication of its two input signals. One example of such an analogue multiplier is the current-mode translinear multiplier formed by transistors M1 to M4 shown in Figure 2. The output current signal (OUT) is proportional to the multiplication of the current signals applied at the IN and WEIGHT (following digital to analogue conversion) terminals.

In an alternative embodiment, the multiplier cell 2 may be a conventional, linear multiplying DAC with reprogrammable weights that are digitally programmed by the microprocessor having a precision and an analogue input signal that scales the output. In another alternative embodiment, a 'rational' multiplying DAC implements both the digital to analogue converter as well as the multiplier cell 2. The M-bit digital inputs of the rational DAC are mapped to two N-bit digital values that comprise the numerator and denominator of the fraction that approximates most closely the digital signal. The two fractional parts are applied to the inputs of two linear conventional N-bit DACs and a multiplier circuit provides the division of the outputs, as well as any required scaling of the analogue input signal, thereby performing both multiplication and division simultaneously. In both alternative embodiments, multiplication is performed by the DAC circuit, obviating the need for a separate multiplier circuit. The precision of the conversion of the M-bit digital weight is proportional to the magnitude of the weights, and its precision may be scaled by first compressing the digital signal at the input to the rational DAC by a compression factor and scaling the analogue input signal by an equivalent scaling factor. This tuning of precision may be performed dynamically. With appropriate tuning of the compression and scaling factors, a precision that is greater than that achieved by a linear conventional DAC of (2N-1) bit precision is achieved. Such a means for variable precision digital to analogue conversion is known as a rational DAC scheme, and is described in more detail below.

With reference again to Figure 1, the outputs of the multiplier cells 2 are applied to respective inputs 5 of a reconfiguration bridge (R-BRIDGE) 6 for controlling an array of *N* multipliers and is the physical mapping of a reconfiguration function to effect the redirection of the output from each of the multipliers to either the system output or for accumulation with the subsequent weight input or multiplier output. Mathematically, the reconfiguration function, R(.) may be expressed as

$$R(y_i)=o_i=\begin{cases}0 & ,\ (y_i \to y_{i+1} \text{ and } r_i=0) \text{ or } (y_i \to w_{i+1} \text{ and } r_i=2)\\ y_i & ,\ r_i=1\end{cases}$$

where the symbol → denotes a redirection for the purpose of accumulation, and the reconfiguration vector, r, defines the state of the switch, where

$$\mathbf{r} = [r_1 \quad r_2 \quad \mathrm{K} \quad r_N]$$

and the value of r_i is 0 when the accumulation of the ith output with the (i + 1)th output is desired, or 1 when the i^{th} output is meant to be expressed as a system output, or is 2 when the accumulation of the i^{th} output with the (i + 1)th weight input is desired.

The R-BRIDGE 6 is illustrated in more detail in Figure 3, and consists of a network of switches controlled by the microprocessor 4. A first set of SELECT switches 7 selectively couple the inputs of the R-Bridge 6 to respective outputs 8 of the R-BRIDGE or to respective weight inputs 10 of the adjacent multiplier, whilst a second set of SUM switches 9 selectively couple neighbouring inputs/outputs of the R-BRIDGE together. The switch network may be controlled by a single reconfiguration vector, r(t).

The functionality of the array of N multiplier cells 3 may be described mathematically as an element-by-element multiplication of two vectors having the same dimensions, which is otherwise known as a Haddamard multiply (denoted by “o”). In the case of vector matrix multiplication, the R-BRIDGE 6 then operates on the Haddamard product to give n dot product outputs, $y_{1 \ldots n}$. Vector matrix multiplication results are obtained from sequential outputs in time. Mathematically,

$$\begin{aligned} \mathbf{y}(t) &= (\mathbf{x}(t) \circ \mathbf{w}(t))^T \bullet \mathbf{s}(t) \\ &= [\mathbf{x}_1(t)^T \bullet \mathbf{w}_1(t) \quad \mathbf{x}_2(t)^T \bullet \mathbf{w}_2(t) \quad \mathrm{K} \quad \mathbf{x}_n(t)^T \bullet \mathbf{w}_n(t)]^T \\ &= [y_1 \quad y_2 \quad \mathrm{K} \quad y_n]^T \end{aligned} \qquad (3)$$

where $\mathbf{x}(t) = [\mathbf{x}_1(t) \quad \mathbf{x}_2(t) \quad \mathrm{K} \quad \mathbf{x}_n(t)]^T \qquad (4)$

$$\mathbf{w}(t) = [\mathbf{w}_1(t) \quad \mathbf{w}_2(t) \quad \mathrm{K} \quad \mathbf{w}_n(t)]^T \qquad (5)$$

and $\mathbf{x}_{1..n}(t)$ and $\mathbf{w}_{1..n}(t)$ are the n inputs and weight vectors respectively, such that the length of $\mathbf{x}_n$ is equal to the length of $\mathbf{w}_n$ and the sum of the lengths of the n input (weights) vectors is equal to the lengths of $\mathbf{x}(t)$ ($\mathbf{w}(t)$). If the bit stream to programme the reconfiguration bridge is a vector, $\mathbf{r}(t)$,

$$\mathbf{r}(t) = [r_1(t) \quad r_2(t) \quad K \quad r_{N-1}(t) \quad 1]^T \tag{6}$$

then the reconfiguration matrix **s**(*t*) is given by

$$\mathrm{s}(t) = \begin{bmatrix} s_{11} & K & s_{1n} \\ M & O & M \\ s_{N1} & \Lambda & s_{Nn} \end{bmatrix} \tag{7}$$

where, $s_{ij} = \begin{cases} 1 & , \sum_{k=0}^{i-1} r_k + 1 = j, \text{where } r_0 = 0 \\ 0 & , \text{otherwise} \end{cases}$, $i = 1..N$, and $j = 1..n$

(8)

In (3), the reconfiguration matrix, **s(t)**, serves to isolate different groups of the Haddamard product, as well as to cumulate the outputs within the group, affording the system a high degree of flexibility and reconfigurability. The reconfiguration bridge implements the necessary mapping of **r(t)** to **s(t)** in (3) and (7).

By operating appropriate switches using a single reconfiguration vector, **r(t)**, which is a single digital word, it is possible to connect together the outputs of all or a subset of the multiplier cells, and to couple the resulting sum of products (Kirchhoff's current law) to a selected output. In (3), the reconfiguration matrix, **s(t)**, serves to isolate different groups of the Haddamard product, as well as to cumulate the outputs within the group, affording the system with a high degree of flexibility and reconfigurability. The reconfiguration bridge implements the necessary mapping of **r(t)** to **s(t)** in (3) and (7).

Thus for example, the outputs of the top three multiplier cells 3 in the arrangement of Figure 3 could be coupled together, and the result output on the upper output of the R-BRIDGE 6. In addition, the outputs of the lower three multiplier cells 3 could be coupled together, and the result output on the lower output of the R-BRIDGE 6. The two sum of product operations are isolated and performed in parallel.

As an example, a 6-dimensional network may be configured to implement a 2×2, a 3×3 VMM and a scalar product, where **r**(t) = $(010011)_b$. On the other hand, if **r**(t) = $(000001)_b$, then a 6×6 VMM operation is implemented.

In very many applications requiring multiplication operations such as vector dot product or VMM, power consumption is a critical factor. This is especially so in the case of portable devices. Another significant consumer of power in an electronic system is often the digital to analogue converters.

Figure 4 illustrates schematically a digital to analogue conversion system suitable for use in a multiplier cell 2 of the VMM of Figure 1. The system is referred to here as a "rational" DAC and comprises an input 11 for receiving an M-bit digital input signal. This digital input signal would be a weight w_k generated by the microcontroller 4. The input signal is applied to a mapping function G(w_k) 12 which uses a look-up table 13 to determine the closest approximation of the value of the digital signal to one of a number of fractions stored within the look-up table. The mapping function then identifies an N-bit numerator and an N-bit denominator which results in the determined fraction. In practice, it is likely that the mapping function G(w_k) will be implemented using a software routine executed by the microprocessor 4, and will not form a physical part of the multiplier cell 2.

The system further comprises two (conventional, linear) N-bit DACs 14,15, to the respective inputs of which are applied the identified N-bit numerator and denominator. The converted analogue outputs appear at analogue outputs of the DACs, and are applied to inputs of a translinear multiplier 16. The translinear multiplier 16 is illustrated in more detail in Figure 5 and comprises four weakly inverted MOS or bipolar junction transistors M1 to M4 having exponential I-V characteristics. The transistors exploit the translinear principle to provide one quadrant multiplication. The multiplier 16 provides at an output a signal Q(.) which is proportional to the ratio of the analogue outputs of the DACs 14,15. A further input of the multiplier 16 receives a control input comprising a current signal I_B which is applied to scale the output signal.

Considering in more detail the translinear multiplier 16, the product of the left-hand currents is equal to the product of the right-hand currents, if the transistors are BJTs or MOSFETs operated in the subthreshold region, and exhibit an exponential characteristic. Hence, the output current of the system having an input *x* may be expressed as:

$$I_{OUT} = Q_R(w_k) = \frac{I_1(Q_{U1}(w_k))}{I_2(Q_{U2}(w_k))} I_B \qquad (9)$$

where I_B is a bias or scaling current and $I_1(.)$ and $I_2(.)$ are the currents due to the uniformly quantised current DACs, $Q_{U1}(.)$ and $Q_{U2}(.)$ respectively. The mapping function *G*(.), which is implemented within the DSP, chooses $Q_{U1}(.)$ and $Q_{U2}(.)$ by finding the closest match in the lookup table.

Two alternative rational DAC current- and voltage-mode implementations are shown in Figures 6 and 7. In all implementations, two low resolution standard linear DACs are employed to provide the necessary ratios.

The operation of the system of Figure 4 is best illustrated by way of example. Assume that M equals 4, and that the mapping function is capable of generating a numerator *u* and a denominator *v* each of 2-bits. Each of these components can have four possible values, i.e. 00, 01, 10, and 11, meaning that *u/v* has eleven possible values, 1/4, 1/3, 1/2, 2/3, 3/4, 1, 4/3, 11/2, 2, 3, and 4. These values are illustrated in Table 1 below. The two DAC system therefore results in a greater number of quantisation states than would be achieved by a single 3-bit linear DAC (having eight possible states), receiving a 3-bit weight w_k from the microprocessor 4.

The quantisation states in the rational DAC scheme proposed here are proportional to the ratios of the quantisation states of two similar linearly quantised DACs, or mathematically:

$$Q_R \propto \frac{Q_{U2}}{Q_{U1}} \qquad (10)$$

where Q_R are the possible quantisation steps of a rational scheme and Q_{U1} and Q_{U2} are those of a uniformly quantised system. Q_{U1} and Q_{U2} have the same precision, which shall be defined as the "base precision" of Q_R. With this scheme, two N-bit DACs may be tuned in tandem to obtain more than 2 raised to the power of (2N-1) quantisation states. A look-up table of the numerator, denominator and the corresponding ratios may be used to match the input to the pair of integer values, which would give the closest approximation. The number of unique quanta, and hence, the size of the look up table for rational DACs having a base precision up to 8-bits is tabulated in Table 2 below.

The quantisation steps produced by a rational DAC scheme are not uniformly sized but rather increase with signal magnitude. For example, Figure 8 shows the possible 159 unique quanta of a rational DAC with a base precision of 4-bits. As the quantisation step is small for small inputs and large for large inputs, the output from a rational DAC suffers substantially less distortion at low input signal values as compared to a purely uniform DAC.

The rational DAC scheme offers the possibility of lower power by reducing full scale currents. The power, P_U, of an N-bit current steering DAC is:

$$\begin{aligned} P_U(N) &= V_{pp} I_u \left(\sum_{n=0}^{N-1} 2^n + 1 \right) \\ &= 2^N V_{pp} I_u = V_{pp} I_{max} \end{aligned} \qquad (11)$$

where V_{pp} is the potential difference across the supplies, I_u is the current at the minimum step size and I_{max} is the full scale current, given as $I_{max} = 2^N I_u$. It can be seen that for a given I_{max}, the power consumption is independent of precision, or alternatively that for a given I_u, I_{max} and P_U are dependent on the precision. Since the rational DAC achieves a higher resolution through low precision uniform DACs, it has the possibility of lower power consumption, while still meeting the physical constraints placed on device sizing, power supply rejection ratio and other parameters.

The performance of a rational DAC has been compared to that of a uniform DAC using MATLAB™. The figures of merit are relative power consumption and percentage distortion error at each quantum.

Power consumption

The input power, P_R, of the rational DAC in Figure 5 is

$$P_R(w_k) = 2P_U(N) + V_{pp}\left(Q_R(w_k) + I_B\right) \qquad (12)$$

where P_U (N) is the input power of an N-bit uniform DAC. A comparison was made between an 8-bit uniform DAC having 256 quantisation levels and a rational DAC with base precision of 4-bits having 159 quantisation levels. The input w_k is a uniformly monotonic vector with 10^4 elements to approximate a continuous signal increasing linearly from 0 to 1. The increasing size of the quantisation steps used with a rational DAC is clearly shown in Figure 10a, which shows the plots of normalised power consumption for both schemes at all quantisation levels. For ease of comparison, the probability of each quantisation step is assumed to be equal and the mean power consumption is indicated by dotted lines for both the uniform and rational DAC schemes, as shown in Figure 10b. As expected, the mean power consumption of the rational DAC is significantly less than that of the uniform DAC. On average, an ideal uniform DAC consumed 6 times the power required for a rational DAC.

Percentage distortion error

The percentage distortion error ε_d at a given quantisation level is defined as the ratio of the difference between the quantised and actual value, to the actual value as a percentage, or mathematically:

$$\varepsilon_d = \frac{w_k - Q(w_k)}{w_k} \times 100\% \,. \qquad (13)$$

For a uniform DAC, the error is attenuating in an inverse manner, as the error remains within a constant band for increasing w_k. The error for the rational scheme is more complex, comprising increasingly wide bands that are attenuating, due to increasing numerator values, within the bands. This is illustrated in Figure 10c. The initial error at the start of each band increases with x due to decreasing denominator values. The bands overlap for small w_k, but become more distinct as w_k increases. Consequently, the output of the rational DAC was significantly less distorted for small values of w_k than that of a linear DAC. However, this was true for less than 24% of the time; for large values of w_k, the distortion due to the rational DAC was much worse than that due to the uniform DAC.

The poor performance of the rational DAC for large values of w_k may be attributed to the skewed distribution of quanta, which is bottom heavy and sparse at the top. For a 4-bit rational DAC, only 8 quanta account for values of w_k from 0.5 to 1, leaving 151 steps for smaller values. By discarding the larger quanta at the top, the overall precision of the rational DAC system may be increased.

Scalable precision rational DACs

Although the performance of the rational scheme is deficient to that of a linear scheme (having the same total input bit precision), its power consumption is significantly less. By trading power consumption for bit precision, the SNR of the rational DAC may be improved. This can be achieved by compressing w_k by a scale factor A so as to "ignore" the quanta at the upper end of the dynamic range. Consequently, the reduced signal is quantised by the finer quanta at the lower end of the quantisation "ladder" to obtain an increase in precision. To obtain the same full scale analogue output signal, the bias current needs to be increased by an equivalent factor of A. Power consumption increases proportionally to SNR. The output current, $Q_R(.)$, of the system having an input, w_k, may be expressed as

$$I_{OUT} = AQ_R(w_k) = \frac{I_1\left(Q_{U1}\left(\frac{w_k}{A}\right)\right)}{I_2\left(Q_{U2}\left(\frac{w_k}{A}\right)\right)} AI_B \qquad (14).$$

Consequently, if A is tuned at run-time, a feasible scheme for scaling the effective precision of the DAC may be achieved, enabling better power management of a device.

The architecture in Figure 5 is modified as shown in Figure 9 to implement the function of scaling precision. The input to the block $G(.)$ is reduced by A within the DSP, while I_B is increased by an equivalent factor. Thus, the expression for the power of a scalable rational DAC is then

$$P_R(w_k) = P_{U1}(w_k) + P_{U2}(w_k) + A\big(I_B + Q_R(w_k)\big)V_{pp}\,. \tag{15}$$

The performance of a scalable precision rational DAC with base precision of 5-bits was compared MATLAB™ to that of a uniform DAC with precision from 5 to 8 bits at different scale factors. As with the above analysis, two figures of merit are tabulated and compared: relative power in percent and relative percentage distortion error. Relative power in percent is calculated as:

$$\text{Rel. Power (\%)} = \frac{\text{mean(rational power)}}{\text{mean(uniform power)}} \times 100\%\,. \tag{16}$$

Relative percentage distortion error (PDE) is the relative measure of the percentage distortion error between rational and uniform DAC, and is calculated as the sum of elements that are described by the greater than, less than or equal operators. For example,

$$(\text{PDE}_\text{R} > \text{PDE}_\text{U})(\%) = \frac{\text{count of}\,(\varepsilon_{d_R} > \varepsilon_{d_U})}{\text{no. of input elements}} \times 100\%\,. \tag{17}$$

The simulation results bear out the analysis that higher precision may be obtained by compressing the inputs such that the signal is quantised within a smaller but denser range and then expanding it to obtain the same full scale range, at the cost of increasing power consumption. However, due to the possibility of a smaller full scale range, the power consumption levels are still significantly less than that which is required for a uniform DAC. For example, the results shown in Table 3 below and plotted in Figure 11 show that at 35% of the input power of an 8-bit uniform DAC, the rational DAC scaled at *A*=10 had a lower percentage distortion error than the uniform DAC for 46.8% of the time and had similar errors for 17.8% of the time.

In general, the effective precision of the quantisation ($\text{PDE}_\text{R} < \text{PDE}_\text{U}$) increases with *A*. This is matched by a comparable increase in power consumption. Hence, it may be observed that *A* has a direct effect on both the precision and power consumption of the DAC. By designing the system so that *A* is readily tuneable, a rational DAC capable of scaling precision at run-time may be implemented. It may be noted that certain values of *A* result in sub-optimal performance, in which a uniform DAC outperforms a rational DAC either in terms of precision of power consumption, or both. Such configurations are denoted with an asterisk "*" in Table 3.

The cost of added software or digital complexity in requiring a look-up table and processing for matching the input to the appropriate index in the LUT has not been addressed in the preceding discussion. While this cost is not insignificant when taken alone, within the context of a mixed-signal processing environment whereby an intimate mix of digital and analogue computation exists, requiring numerous low-power, local DACs of reasonable performance, the cost of digital complexity may be shared by all the DACs. Thus, with a large number of local DACs, the cost of this added digital complexity becomes very low. This is true in the VMM described above.

Rational DACs implement a non-linear quantisation scheme, and it is appropriate to compare it with a logarithmic scheme with quantisation steps, $Q_{LOG}(.)$ given as:

$$Q_{LOG}(w) = r^{b_0 2^0 + b_1 2^1 + \ldots + b_{N-1} 2^{N-1}} . \qquad (18)$$

If $k = b_0 2^0 + b_1 2^1 + \ldots + b_{N-1} 2^{N-1}$, then the maximum distortion error occurs midway between two adjacent quantisation steps, and is given as

$$\varepsilon_d = \frac{\frac{r^k + r^{k-1}}{2} - r^k}{\frac{r^k + r^{k-1}}{2}} = \frac{1-r}{1+r} \cdot \qquad (19)$$

It is clear that the maximum distortion error is independent of k and hence, is constant for all input values (it may also be noted that maximum distortion error is independent of the number of bits used). Thus, with such a scheme, the technique of companding as described above may not be used to effectively scale the SNR of the DAC.

To improve SNR performance, r must be set closer to unity, imposing stricter limits during fabrication. An increase in r is matched by a corresponding reduction in the dynamic range, δ_L, of the DAC, since $\delta_L = r^{-2^{N-1}}$. Therefore, to improve SNR without loss of dynamic range, stricter constraints on r as well as a need for more bits are required. In contrast, the SNR of the proposed rational DAC scheme may be readily tuned at run-time. Furthermore, the rational DAC has a wide dynamic range, δ_R, which is defined as the largest to the smallest number represented, and is given as $\delta_R = 2^N/2^{-N} = 2^{2N}$ for a rational DAC with base precision of *N*-bits.

A vector dot product multiplier has been described above with reference to Figure 1. A modified arrangement of this architecture for use in evaluating polynomial functions will now be described.

Polynomial functions form an important class of multiply and accumulate functions, and their usefulness can be seen from the diverse applications where they are found, such as in power series for solving differential equations and in the approximating of analytical functions. Polynomial functions may be defined as follows:

$$y = \sum_{m=0}^{N-1} a_m x^m = a_{N-1}x^{N-1} + a_{N-2}x^{N-2} + \mathrm{K} + a_1 x + a_0 \qquad (21)$$

A naive analysis of this equation might lead to the conclusion that

$$(N-1)+(N-2)+\mathrm{K}+2+1 = \tfrac{1}{2}\left(N^2 - N\right) \qquad (22)$$

multiplications are required to compute the output. However, by using Horner's rule (J.-M. Muller, "Algorithms and Architectures," in Elementary Functions. Boston: Birkhauser, 1997, pp. 43), the equation may be factorised to:

$$y = \left(\left(a_{N-1}x + a_{N-2}\right)x + \mathrm{K}\right)x + a_0 \qquad (23)$$

which requires only *N-1* multiplications and additions as a consequence of the polynomial function being expressed as a series of nested linear equations.

With reference to Figure 12, by incorporating "feedforward" between the outputs of the reconfiguration bridge 6 and summing elements (not shown but present at the junctions between the weight input lines and the feedback lines), the structure can be configured to perform not only vector dot product multiplication, but also to evaluate polynomial functions. In Figure 12, all of the lines shown carry analogue signals and all of the processing is carried out in the analogue domain. Digital to analogue conversion of the digital weights is carried out by respective DACs, not shown in the Figure. It will however be appreciated that multiplying DACs (including rational DACs) may be used, with both the digital weights and analogue feedback signals being supplied to the multiplying DACs. This would require a modification to the multiplying DACs as described above.

In Figure 12, the outputs from adjacent multiplier pairs used to represent *y* = *ax* + *b* linear equations are fed to the subsequent weight input of the next pair, resulting in a nested representation of a polynomial function factorised using Horner's Rule. The resulting expression in terms of the network variables is:

$$y = w_1 x^{M} + \sum_{m=1}^{M} w_{2m} x^{M-m} \qquad (24)$$

where $M = \mathrm{floor}\left(\frac{N}{2}\right)$.

Consider now the specific example of a third order polynomial:

$$\begin{aligned} f(x) &= a_1 x^3 + a_2 x^2 + a_3 x + a_4 \\ &= \left(\left(a_1 x + a_2\right)x + a_3\right)x + a_4 \end{aligned} \qquad (25)$$

To map this to the proposed network, we would need to have the inputs such that the signal input array of Figure 12 has the following values: **x** = [*x* 1 *x* 1 *x* 1] and the (digital) weights have the following values: **w** = [a_1 a_2 0 a_3 0 a_4]. This is illustrated in Figure 13. If we label the multiplication elements MUX1-6, Table 4 below illustrates the respective inputs and outputs, from which it can be seen that multiply and accumulate (MAC) terms, i.e. a sum of a linear component and a constant, of the factorised equation are initially evaluated, and the results fed forward to allow evaluation of higher order terms.

The reconfiguration function, *R*(.), as described above is realised using a network of switches within the reconfiguration bridge 6, with the purpose of isolating or cumulating (summing) outputs from the multiplier array, depending on the reconfiguration required. Each multiplier output in the array may be connected to a two-way toggle, as shown in Figure 14a, to implement the reconfiguration function. The input current, I_{IN}, is directed to outputs, $OUT1$ or $OUT2$, depending on the potential of TG, where

$$V_{TG} = \begin{cases} +V & ,r=1 \\ -V & ,r=2 \end{cases}$$

When TG is high, the multiplier output is shorted to the next multiplier output, i.e. $y_i \rightarrow y_{i+1}$, except for the last multiplier output, y_N, which is shorted to the output, o_N, for all instances. A short between the multiplier output, y_i, and the output, o_i, occurs also if TG is low.

With the reconfiguration function $R(.)$ above, a two-way toggle is needed for all even outputs, while a three-way toggle, as shown in Figure 14b, is needed for each of the odd-indexed multiplier outputs, with the exception of the last multiplier output which is shorted to the output. The three-way toggle comprises two cascaded two-way toggles. The output, OUT2 of the two-way toggle is fed to the input of another two-way toggle to realise a three-way toggle. In both the two- and three-way toggles, the flow of current to the outputs requires a potential gradient between the input and outputs.

The switches shown in Figure 14 may be implemented using NMOS and PMOS transistors in a pass transistor fashion for simplicity although, in practise, such a realisation would see a loss of voltage "head-room" with each cascade as a result of the need for the potential difference across the drain and source of the transistor to be greater than the threshold voltage of that device. In a practical implementation, transmission gates may be used to overcome this limitation.

It will be appreciated by the person of skill in the art that variations may be made to the above described embodiments without departing from the scope of the present invention.

Word	Value	Ratio
0000	0	OFF
0001	1/16	00/11
0010	1/8	00/01, 01/11
0011	3/16	10/11
0100	1/4	00/00, 01/01, 10/10, 11/11
0101	5/16	11/10
0110	3/8	10/01
0111	7/16	10/01
1000	1/2	01/00, 11/01
1001	9/16	01/00, 11/01
1010	5/8	01/00, 11/01
1011	11/16	10/00
1100	12/16	10/00
1101	13/16	10/00
1110	7/8	11/00
1111	15/16	11/00

Table 1

Base precision of $Q_R(.)$	No. of quanta	Equivalent no. of quanta in bits
1	3	>1
2	11	>3
3	43	>5
4	159	>7
5	647	>9
6	2,519	>11
7	10,043	>13
8	39,895	>15

Table 2

Precision, N	Scale, A	Relative power	$PDE_R < PDE_U$	$PDE_R = PDE_U$	$PDE_R > PDE_U$
5	1*	245.5	35.7	64.3	0
6	1*	124.6	24.4	50.6	25.0
7	1	62.8	19.1	32.6	48.4
	2	63.6	30.0	27.2	42.8
	3	64.3	32.6	35.7	31.7
	4	65.1	44.6	22.1	33.3
	5	65.9	45.8	27.8	26.4
8	1*	31.5	15.0	19.9	65.1
	2*	31.9	23.4	17.4	59.2
	3*	32.3	25.4	24.5	50.1
	4*	32.7	34.2	15.5	50.4
	5	33.1	34.7	21.4	43.9
	6	33.5	37.7	20.7	41.7
	7	33.9	41.4	18.9	39.7
	8	34.2	46.0	12.7	41.3
	9	34.6	43.0	21.9	35.1
	10	35.0	46.8	17.8	35.5

Table 3

	IN1	IN2	OUT
MUX1	x	a_1	a_1x
MUX2	1	a_2	a_2
MUX3	x	$0 + a_1x + a_2$	$(a_1x + a_2)x$
MUX4	1	a_3	$(a_1 + x)x + a_3$
MUX5	x	$0 + ((a_1 + x)x + a_3)$	$((a_1 + x)x + a_3)x$
MUX6	1	a_4	$((a_1 + x)x + a_3)x + a_4$

Table 4